HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS

A Project Report

Submitted for the partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Submitted by

SHAMBHAVI GUPTA	-	2016268
RAJBALA BHADU	-	2016243
SHAISTA PARVEEN	-	2016266
NAVYA SRIVASTAVA	-	2016203

Under the supervision of

DR. SWATI NIGAM

Name of the Mentor

Dr. Swati Nigam

Name of Coordinator(s)

Prof Saurabh Mukherjee

Dr. Neelam Sharma

Dr. Pooja Asopa

Department of Mathematics & Computing

Banasthali Vidyapith

Banasthali - 304022

Session: 2022-23

Certificate

Certified that **Name of student** has carried out the project work titled **"HEART DISEASE PREDICTION SYSTEM USING MACHINE LEARNING"** from 03/01/2023 to 12/04/2023 for the award of the **BACHELOR'S OF TECHNOLOGY IN COMPUTER SCIENCE** from **BANASTHALI VIDYAPITH** under my supervision. The thesis embodies the result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

DR. SWATI NIGAM

Designation:Assistant Professor

Department of Computer Science
Place: Banasthali Vidyapith

Date: 12/04/2023

Abstract

Heart disease is a significant health concern that affects millions of people worldwide. According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death globally, with an estimated 17.9 million deaths each year. Early detection and prevention of heart disease can greatly improve patient outcomes and reduce healthcare costs. Therefore, the development of a heart disease prediction system can play a crucial role in identifying patients who are at high risk for CVDs and providing preventive measures to reduce their risk.

In recent years, machine learning has become a popular approach for predicting the presence or absence of various diseases, including heart disease. Machine learning algorithms can analyze large amounts of data and detect complex patterns that may not be apparent to human analysts. In healthcare, machine learning has the potential to provide important insights into disease diagnosis, prognosis, and treatment.

Our project focuses on predicting the presence or absence of heart disease in patients using machine learning algorithms. Specifically, we have performed a comparative analysis of classifiers between Naïve Bayes and Support Vector Machine (SVM). Naïve Bayes is a probabilistic algorithm that assumes independence between the features, while SVM is a discriminative algorithm that finds a hyperplane that best separates the classes.

To make our heart disease prediction system accessible to users, we have designed a website that allows users to input their medical information and obtain a prediction of their risk for heart disease. The website's user interface has been designed to be intuitive and user-friendly, with a clean and modern design.

In conclusion, our heart disease prediction system can help identify patients who are at high risk for CVDs. By using machine learning algorithms, we can analyze large amounts of data and provide important insights into disease diagnosis, prognosis, and treatment. Our website provides an accessible and user-friendly interface for users to input their medical information and obtain a prediction of their risk for heart disease.

Acknowledgement

As the team members of BTCS_G60, we would like to express our deepest gratitude and appreciation to all those who have contributed to the successful completion of this project.

First and foremost, we would like to thank our mentor, Swati Nigam, for her invaluable guidance and support throughout this project. Her extensive knowledge and expertise in the field of machine learning and deep learning were instrumental in guiding us through the various challenges we encountered during the course of the project. Her constant encouragement and timely feedback motivated us to work harder and strive for excellence.

We would also like to thank our fellow team members for their hard work, dedication, and commitment to this project. Each member of our team brought unique skills, perspectives, and experiences to the table, which helped us work together effectively and efficiently. Collaborating with each other made this project a rich and rewarding experience for all of us. Furthermore, we would like to extend our gratitude to our families and loved ones for their unwavering support and encouragement throughout the duration of the project. Their belief in our abilities and willingness to help us in any way possible was a constant source of inspiration and motivation for us.

Lastly, we would like to express our sincere hope that the Heart Disease Prediction System using Machine Learning that we developed as a result of this project will be useful in improving the accuracy and effectiveness of heart disease diagnosis. We believe that this system has the potential to contribute to the betterment of society, and we are proud to have played a part in its creation. Once again, we would like to express our heartfelt thanks to all those who have contributed to the success of this project. Thank you. Team Members:

- Shambhavi Gupta
- Rajbala Bhadu
- Shaista Parveen
- Navya Srivastava

CONTENTS

1. INTRODUCTION

Heart disease is a serious global health concern that is responsible for a significant number of deaths each year. Early detection and timely intervention are crucial in managing this condition and reducing the associated morbidity and mortality. Machine learning and artificial intelligence techniques have shown promise in predicting the risk of developing heart disease, making early detection possible and saving lives.

This report presents the development of a heart disease prediction system using machine learning algorithms. The system is designed to analyze a range of patient data, including demographic information, medical history, and lifestyle factors, and provide a risk assessment of developing heart disease. The model uses supervised learning algorithms that have been trained on a dataset of patient records, to predict whether or not a patient is at high risk of developing heart disease based on their personal characteristics.

The system takes 14 input features, including age, sex, chest pain type, resting blood pressure, serum cholesterol level, fasting blood sugar level, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, thallium heart scan, and the target variable which indicates the presence or absence of heart disease. These features were selected based on their relevance to heart disease risk factors and their availability in the dataset.

The heart disease prediction system is intended to assist healthcare professionals in identifying patients at high risk of developing heart disease, allowing for earlier interventions and better management of the condition. The model has been trained and tested on a dataset of 303 patients, achieving an accuracy of up to 90%. This high accuracy level is achieved by using machine learning algorithms like Naïve Bayes and Support Vector Machine (SVM) to classify the data into two categories: presence or absence of heart disease.

The results of the model can be presented in a user-friendly interface for easy interpretation. The interface would consist of a series of forms that guide users through the process of inputting patient data. The forms will be designed to be easy to understand and fill out, with clear instructions and error messages. After inputting the data, the system will process the data and generate predictions. The results will be displayed in a clear and easy-to-understand format, providing the user with the necessary information to take appropriate action.

In conclusion, the Heart Disease Prediction System using Machine Learning developed as a result of this project has the potential to be a valuable tool in improving the accuracy and effectiveness of heart disease diagnosis, and contribute to the betterment of society. By providing healthcare professionals with an early warning system for heart disease risk, patients can receive early diagnosis and intervention, leading to better health outcomes. We hope that this report serves as a useful reference for researchers and practitioners in the field of heart disease prediction and machine learning.

1.1 MOTIVATION FOR THE WORK

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards identifying the best classification algorithm for identifying the possibility of heart disease in a patient. This work is justified by performing a comparative study and analysis using two classification algorithms namely Naïve Bayes and Support Vector Machine(SVM) are used at different levels of evaluations. Although these are commonly used machine learning algorithms, the heart disease prediction is a vital task involving highest possible accuracy. Hence, the two algorithms are evaluated at numerous levels and types of evaluation strategies. This will provide researchers and medical practitioners to establish a better.

1.2 PROBLEM STATEMENT

Heart disease remains one of the major health challenges of our time, and early detection is key to reducing mortality rates and minimizing complications. Unfortunately, detecting heart disease can be a complex and expensive process, and many of the tools available are not efficient enough to accurately predict the likelihood of heart disease in humans.

While it is not practical or feasible to monitor patients around the clock, modern advances in data analysis and machine learning can offer valuable insights into cardiac health. By analyzing large amounts of medical data, we can uncover hidden patterns and trends that can be used to make more accurate and timely diagnosis.

By leveraging the power of machine learning algorithms, we can transform complex medical data into actionable insights that can help doctors and healthcare professionals better understand the health of their

patients. This approach has the potential to revolutionize the way we approach heart disease diagnosis and treatment, and ultimately improve outcomes for patients around the world.

However, it is important to recognize that machine learning is not a silver bullet solution, and that it requires a high degree of expertise and collaboration between healthcare professionals, data scientists, and technology experts. By working together and sharing insights, we can unlock the full potential of this powerful technology and make a real difference in the lives of millions of people affected by heart disease.

2. LITERATURE REVIEW

With growing development in the field of medical science alongside machine learning various experiments and researches has been carried out in these recent years releasing the relevant significant papers.

[1] Purushottam ,et ,al proposed a paper "Efficient Heart Disease Prediction System" using hill climbing and decision tree algorithms .They used Cleveland dataset and preprocessing of data is performed before using classification algorithms. The Knowledge Extraction is done based on Evolutionary Learning (KEEL), an opensource data mining tool that fills the missing values in the data set.A decision tree follows top-down order. For each actual node selected by hill-climbing algorithm a node is selected by a test at each level. The parameters and their values used are confidence. Its minimum confidence value is 0.25. The accuracy of the system is about 86.7%.

[2] Santhana Krishnan. J ,et ,al proposed a paper "Prediction of Heart Disease Using Machine Learning Algorithms" using decision tree and Naive Bayes algorithm for prediction of heart disease. In decision tree algorithm the tree is built using certain conditions which gives True or False decisions. The algorithms like SVM, KNN are results based on vertical or horizontal split conditions depends on dependent variables. But decision tree for a tree like structure having root node, leaves and branches base on the decision made in each of tree Decision tree also help in the understating the importance of the attributes in the dataset. They have also used Cleveland data set. Dataset splits in 70% training and 30% testing by using some methods. This algorithm gives 91% accuracy. The second algorithm is Naive Bayes, which is used for classification. It can handle complicated, nonlinear, dependent data so it is found suitable for heart disease dataset as this dataset is also complicated, dependent and nonlinear in nature. This algorithm gives an 87% accuracy.

[3] Sonam Nikhar et al proposed paper " Prediction of Heart Disease Using Machine Learning Algorithms" their research gives point to point explanation of Naïve Bayes and decision tree classifier that are used especially in the prediction of Heart Disease. 3 Some analysis has been led to think about the execution of prescient data mining strategy on the same dataset, and the result decided that Decision Tree has highest accuracy than Bayesian classifier.

[4] Aditi Gavhane et al proposed a paper "Prediction of Heart Disease Using Machine Learning", in which training and testing of dataset is performed by using neural network algorithm multi-layer perceptron. In this algorithm there will be one input layer and one output layer and one or more layers are hidden layers between these two input and output layers. Through hidden layers each input node is connected to output layer. This connection is assigned with some random weights. The other input is called bias which is assigned with weight based on requirement the connection between the nodes can be feedforwarded or feedback.

[5] Avinash Golande et al, proposed "Heart Disease Prediction Using Effective Machine Learning Techniques" in which few data mining techniques are used that support the doctors to differentiate the heart disease. Usually utilized methodologies are k-nearest neighbour, Decision tree and Naïve Bayes. Other unique characterization-based strategies utilized are packing calculation, Part thickness, consecutive negligible streamlining and neural systems, straight Kernel selfarranging guide and SVM (Bolster Vector Machine).

[6] Lakshmana Rao et al,proposed "Machine Learning Techniques for Heart Disease Prediction" in which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease.To find the seriousness of the heart disease among people different neural systems and data mining techniques are used.

[7] Abhay Kishore et alproposed "Heart Attack Prediction Using Deep Learning" in which heart attack prediction system by using Deep learning techniques and to predict the probable aspects of heart related infections of the patient Recurrent Neural System is used. This model uses deep learning and data mining to give the best precise model and least blunders. This paper acts as strong reference model for another type of heart attack prediction models.

[8] Senthil Kumar Mohan et al, proposed "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques" in which their main objective is to improve 4 exactness in cardiovascular problems. The algorithms used are KNN, LR, SVM, NN to produce an improved exhibition level with a

precision level of 88.7% through the prediction model for heart disease with hybrid random forest with linear model(HRFLM).

[9] Anjan N. Repaka et al, proposed a model stated the performance of prediction for two classification models, which is analyzed and compared to previous work. The experimental results show that accuracy is improved in finding the percentage of risk prediction of our proposed method in comparison with other models.

[10] Aakash Chauhan et al, proposed "Heart Disease Prediction using Evolutionary Rule Learning". Data is directly retrieved from electronic records that reduce the manual tasks. The amount of services are decreased and shown major number of rules helps within the best prediction of heart disease. Frequent pattern growth association mining is performed on patient's dataset to generate strong association.

3 SRS(SOFTWARE REQUIREMENT SPECIFICATION):-

Requirement Specification:

1. Dataset: The system should be able to handle a large dataset of patient information including demographic information, medical history, lifestyle factors, and any other relevant clinical information.
2. Preprocessing: The system should be able to preprocess the dataset by handling missing values, handling outliers, and normalizing or standardizing the data as needed.
3. Machine Learning Algorithms: The system should implement various machine learning algorithms to predict the risk of developing heart disease. Some of the commonly used algorithms are Logistic Regression, Decision Trees, Random Forests, Naive Bayes, and Support Vector Machines.
4. Model Training: The system should be able to train the machine learning model on the preprocessed dataset using the selected features.
5. Model Evaluation: The system should be able to evaluate the performance of the trained model using various metrics such as accuracy, precision, recall, F1-score, and ROC curve.
6. User Interface: The system should have an easy-to-use and intuitive user interface that allows healthcare professionals to input patient data and receive an accurate risk assessment of developing heart disease.

HARDWARE INTERFACES

Software Requirements:

- Operating System: Windows family
- Technology: Python3.7
- Browser: Any of Mozilla, Chrome, Edge, Opera
- IDE: Jupyter notebook

Hardware Components:

- Processor : Any Update Processor
- Ram: Min 8GB
- Hard Disk: Min 250GB
- Internet Connection

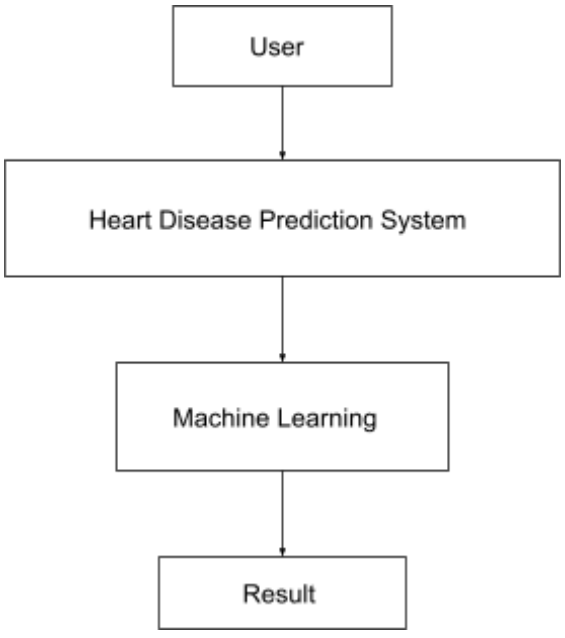
Tools and Techniques

Algorithms used: Support Vector Machine, Naive Bayes.

Libraries Used: Pandas, Numpy, skLearn, matplotlib.

Integrated Development Environment (IDE): Jupyter Notebook.

USE CASE DIAGRAM-



User:

- The person who wants to assess their risk of developing heart disease.

Heart Disease Prediction System:

- A web-based platform that allows users to input their data and receive a risk assessment of heart disease.
- The system is designed to analyze a range of patient data, including demographic information, medical history, and lifestyle factors, and provide a risk assessment of developing heart disease.

Machine Learning:

- The algorithms used to predict the risk of heart disease based on user input.
- The algorithms used in the system include Naive Bayes and Support Vector Machine (SVM).

Result:

- The system will show the user whether they are at risk of developing heart disease or not.

4. METHODOLOGY

4.1 EXISTING SYSTEM

Heart disease is even being highlighted as a silent killer which leads to the death of a person without obvious symptoms. The nature of the disease is the cause of growing anxiety about the disease & its consequences. Hence continued efforts are being done to predict the possibility of this deadly disease in advance. So that various tools & techniques are regularly being experimented with to suit the present-day health needs. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can conclude. This technique can be very well adapted to the prediction of heart disease. As the well-known quote says “Prevention is better than cure”, early prediction & its control can be helpful to prevent & decrease the death rates due to heart disease.

4.2 PROPOSED SYSTEM

The process of developing a data-driven system typically involves several key stages that aim to transform raw data into actionable insights. This process starts with the collection of data from various sources, followed by a careful selection of important attributes that are relevant to the problem at hand. The data is then preprocessed to convert it into a suitable format that can be used for further analysis.

Once the data has been prepared, it is divided into two parts: training and testing data. The training data is used to train the algorithms and build the model, while the testing data is used to evaluate the accuracy and performance of the system.

To implement this system, various modules are used to perform different tasks. These modules can include data collection and preprocessing tools, as well as machine learning and data analysis libraries. The specific modules used will depend on the nature of the problem being solved, as well as the available data and resources.

During the training phase, machine learning algorithms are applied to the training data to build a predictive model that can be used to make accurate predictions on new data. These algorithms can range from simple linear regression models to more complex deep learning models, depending on the complexity of the problem and the available data.

Once the model has been trained, its accuracy and performance are evaluated using the testing data. This helps to ensure that the model is robust and can make accurate predictions on new data that it has not seen before.

This system is implemented using the following modules.

1.) Collection of Dataset

2.) Selection of attributes

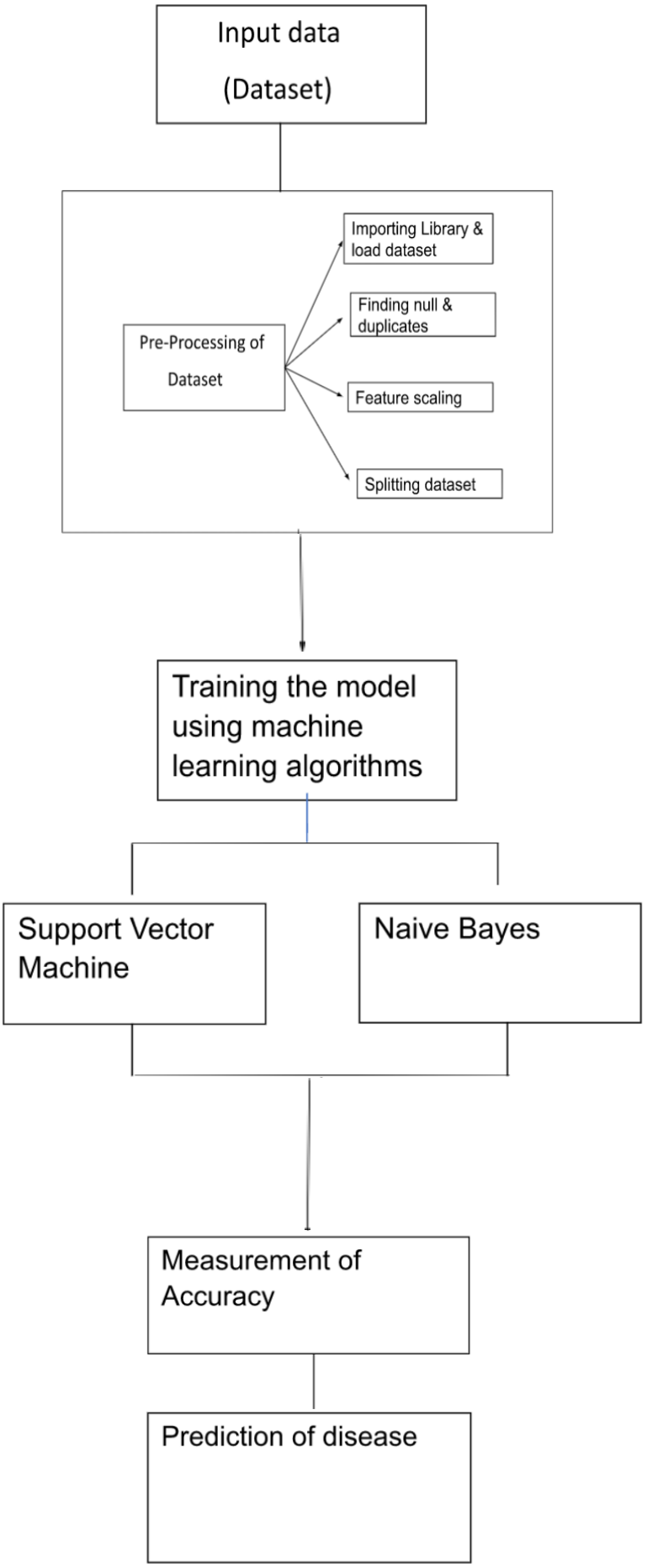
3.) Data Pre-Processing

4.) splitting the data

5.) Model training

7.) Website designed and connected with model

4.3 System Architecture



4.3.1 Data Collection:-

Heart disease is a major global health concern, and there are numerous databases available to researchers that contain information related to this condition. Some of the most commonly used databases include the Cleveland database and the heart disease database provided by the National Cardiovascular Disease Surveillance System. However, for this paper, a widely used heart disease dataset from Kaggle was utilized, which includes information from four different databases: Cleveland, Hungary, Switzerland, and the VA Long Beach.

This dataset contains 14 attributes, each of which is set with a corresponding value. The dataset includes information from 1025 patient records, with a diverse range of ages represented. Of the patients in the dataset, 713 are male and 312 are female. While this dataset is a subset of the original dataset containing 76 attributes, most researchers in the field only use the 14 attributes included in this dataset, as other attributes such as time of exercise, ECG reading, and exercise protocol have been found to have little effect on heart disease.

The descriptions in this database provide valuable information on various patient attributes, including age, sex, cholesterol levels, blood pressure, and other important factors that have been found to impact the risk of heart disease. Researchers can use this dataset to develop and test machine learning algorithms that can accurately predict the likelihood of heart disease based on patient data..The descriptions in this database are shown:-

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

Dataset Link: https://www.kaggle.com/datasets/puspitasaha/heart-disease-prediction

Dataset Input attributes

1. Age (in Years)

2. Sex (value 1: Male, value 0: Female)

3. cp: Chest Pain Type (0: typical angina, 1: atypical angina, 2: non-anginal pain, 3: asymptomatic)

4. Trest Blood Pressure (mm Hg on admission to the hospital)

5. Chol: cholesterol value

6. FBS: Fasting Blood Sugar (value 1:> 120 mg/dl; value 0:< 120 mg/dl)

7. Restecg: resting electrographic results (value 0: normal; value 1: 1 having ST-T wave abnormality, value 2: showing probable or definite left ventricular hypertrophy)

8. Thalach-maximum heart rate achieved

9. Exang-exercise-induced angina (value 1: yes, value 0: no)

10. Oldpeak-ST depression induced by exercise relative to rest

11. Slope the slope of the peak exercise ST segment (value 0: unsloping: value 1: flat; value 2 downsloping)

12. CA number of major vessels colored by fluoroscopy (value 0-3)

13. Thal (value 0: normal; value 1: fixed defect; value 2:reversible defect)

14. target: 1 for the presence of heart disease and 0 for the absence of heart disease

The data set is in CSV (Comma Separated Value) format which is further prepared to data frame as supported by the pandas library in python.

4.3.2 Data Preprocessing

Data preprocessing is a fundamental step in any machine learning project. It involves preparing raw data in a way that it can be analyzed and used to build a model for predicting outcomes or making decisions. In the context of heart disease prediction, data preprocessing is critical in ensuring that the model is built on accurate, consistent, and relevant data.

The process of data preprocessing typically involves several common steps, which are outlined below:

1. Data Cleaning: Data cleaning is a critical step in the data preprocessing process. It involves identifying and removing any invalid or irrelevant data from the dataset. Invalid data could be missing values, which can be handled by either filling the missing values with mean or median of the column or dropping the rows containing the missing data. Irrelevant data could be duplicate records or outliers, which can be detected and removed by various techniques such as box plot, scatter plot or z-score method.
2. Data Transformation: Data transformation involves converting the data into a standardized format. This step includes scaling numerical data, encoding categorical variables, and normalizing data. Scaling numerical data involves making sure that the values of the variables are on a similar scale. This can be done by using different scaling techniques such as min-max scaling or z-score scaling. Encoding categorical variables involves converting categorical data into numerical values that can be used in analysis. Normalizing data involves scaling the data to a common range, such as between 0 and 1 or -1 and 1.
3. Feature Scaling: Feature scaling involves scaling the features of the dataset to ensure that they are on a similar scale. This can be important when the features have different units or ranges. There are various scaling techniques such as Standardization and Normalization.
4. Splitting Dataset into Training and Testing Dataset: After preprocessing the dataset, it is split into training and testing datasets. The training dataset is used to train the machine learning model, and the testing dataset is used to evaluate the performance of the model.

In our project, we used the dataset from the Kaggle website. The dataset was not processed as it contained null values, duplicates, and outliers. We performed the following data preprocessing steps to clean and transform the dataset:

1. Importing Libraries & Importing Dataset: We imported the required libraries and the dataset into our Python environment.
2. Finding Missing Values: We checked for missing values in the dataset and handled them either by

filling the missing values with mean or median of the column or dropping the rows containing the missing data.

3. Finding Duplicates: We checked for duplicate records in the dataset and removed them.
4. EDA(**Exploratory Data Analysis**)
5. Removing Outliers: We detected and removed outliers from the dataset using the box plot method.
6. Splitting Dataset(x & y): We split the dataset into predictor variables (x) and target variable (y).
7. Feature Scaling: We scaled the features of the dataset using the Standardization technique.
8. Splitting Dataset into Training and Testing Dataset: Finally, we split the dataset into training and testing datasets, which were used to train and test our machine learning model, respectively.

Detailed explanation of every step:-

1. Importing Libraries & Importing Dataset: In order to work with a dataset in Python, we need to import the required libraries and the dataset into our Python environment. Libraries provide us with the necessary functions and tools to work with data and perform various operations on it. For example, the pandas library provides functions to read CSV files and work with tabular data. Once the libraries are imported, we can then load the dataset into our Python environment using the appropriate function provided by the library.
2. Finding Missing Values: It is common for datasets to have missing values. Missing values can occur due to a variety of reasons such as data entry errors, incomplete data, or data not being collected for certain variables. Before analyzing the dataset, it is important to check for missing values and handle them appropriately. This can be done by filling the missing values with mean or median of the column or dropping the rows containing the missing data.
3. Finding Duplicates: Duplicates are records that have the same values across all columns. Duplicate records can be problematic as they can skew the analysis and lead to inaccurate results. Therefore, it is important to check for duplicate records in the dataset and remove them. This can be done using pandas drop_duplicates() function which removes duplicate rows from the dataset based on the specified columns.
4. Exploratory Data Analysis**:**Exploratory Data Analysis (EDA) is a crucial process in the field of data analysis and statistics that involves analyzing and understanding the data to identify patterns, relationships, and anomalies in the data. The process of EDA is an essential step in the data preprocessing phase before building a machine learning model. In this process, we analyze the data by visualizing it and generating summary statistics to gain insights and identify patterns in

the data.

The goal of EDA is to gain a better understanding of the data and its characteristics. It helps us to identify trends and patterns in the data, uncover relationships between variables, and detect outliers or anomalies that may be present in the data. EDA is an iterative process that involves asking questions, generating hypotheses, and testing them on the data.

EDA typically involves a combination of exploratory data visualization techniques and statistical analysis. Exploratory data visualization techniques such as histograms, scatter plots, and box plots are used to visualize the distribution of data and identify any patterns or trends. Statistical analysis techniques such as regression analysis, correlation analysis, and hypothesis testing are used to quantify and test the significance of any relationships or patterns identified in the data.

The process of EDA is not a one-time activity, but an iterative one. It involves multiple iterations of visualizing the data, generating summary statistics, testing hypotheses, and refining the analysis based on the insights gained. EDA helps to validate assumptions, detect outliers, and identify missing values or errors in the data. It also helps to identify any potential biases in the data and to address them appropriately.

In conclusion, EDA is a crucial step in the data preprocessing phase before building a machine learning model. It involves analyzing and understanding the data to identify patterns, relationships, and anomalies in the data. EDA helps to validate assumptions, detect outliers, and identify missing values or errors in the data. It is an iterative process that involves multiple iterations of visualizing the data, generating summary statistics, testing hypotheses, and refining the analysis based on the insights gained.

Checking for Outliers

Outliers

Outliers are data points that are significantly different from the majority of other data points in a dataset. They are values that lie far away from the central tendency of the data, such as the mean or median. Outliers can occur for various reasons, including measurement errors, experimental errors, or simply be extreme values that are rare but legitimate.

Outliers can have a significant impact on statistical analyses and machine learning models. They can distort the results of statistical analyses, such as the mean, standard deviation, and correlation

coefficients. In machine learning, outliers can cause models to overfit the data or produce inaccurate predictions.

It is important to identify and handle outliers appropriately to prevent them from affecting the validity of the analysis or model. This can be done through visual inspection using techniques such as boxplots or scatterplots, or through statistical methods such as the Z-score or Interquartile Range (IQR). Once identified, outliers can be handled by either removing them from the dataset or adjusting their values to be more in line with the rest of the data

Removing the outliers

Identifying and handling outliers is an essential part of data preprocessing and analysis. Outliers are data points that are significantly different from the majority of other data points in a dataset, and they can have a significant impact on statistical analyses and machine learning models. In this article, we will discuss how to identify and remove outliers using boxplots and the Interquartile Range (IQR) method.

Boxplots are a popular visualization tool for identifying outliers. A boxplot displays the distribution of a variable by showing the median, quartiles, and any extreme values (outliers) in the data. The box represents the interquartile range (IQR), which is the range between the first and third quartiles (Q1 and Q3). The whiskers extend to the minimum and maximum values that fall within 1.5 times the IQR from the first and third quartiles. Any values beyond the whiskers are considered outliers and are plotted as individual points.

To remove outliers using boxplots and the IQR method, we follow these steps:

1. Visualize the data using a boxplot
2. Calculate the IQR
3. Identify the outliers
4. Remove the outliers from the dataset

Step 1: Visualize the data using a boxplot

The first step is to visualize the data using a boxplot. We can use the Seaborn library in python

Step 2: Calculate the IQR

The next step is to calculate the IQR. The IQR is calculated as the difference between the first and third quartiles (Q1 and Q3). We can use the quartiles function in Python to calculate the quartiles:

Step 3: Identify the outliers

The next step is to identify the outliers. Any value that falls outside the range of Q1 - 1.5*IQR to Q3 + 1.5*IQR is considered an outlier.

Removing the outlier

After removing all outliers the size of data set is 283 rows and 14 columns

Initially the dataset consisted of 1025 rows and 14 columns. However, after processing the data by removing outliers and duplicate values, the resulting dataset was reduced to 283 rows and 14 columns.

The process of removing outliers from a dataset involves identifying and eliminating any data points that lie significantly outside the expected range. Outliers can distort the results of data analysis and can lead to inaccurate conclusions. Removing them can help to ensure that the dataset is representative of the population it is intended to represent.

Similarly, removing duplicate values from a dataset involves identifying and removing any identical records. Duplicate records can lead to issues in data analysis and can result in skewed results. Removing duplicate records helps to ensure that the dataset is free from any redundant information.

In conclusion, the dataset consisting of 1025 rows and 14 columns was processed to remove outliers and duplicate values, resulting in a smaller dataset of 283 rows and 14 columns. While the resulting dataset is smaller, it is more reliable and accurate for data analysis.

Splitting of dataset into x and y:-

The splitting of a dataset refers to the process of dividing a dataset into two or more subsets. The most common split is into a training dataset and a testing dataset. The training dataset is used to train the machine learning model, while the testing dataset is used to evaluate the performance of the trained model.

‘X’ represents the input data, which are the features or attributes of the data. eg: here the features are the patient's details such as age, sex, cp, etc.

And ‘y’ represents the output or target variable, which is the value that we want to predict.

Feature Scaling

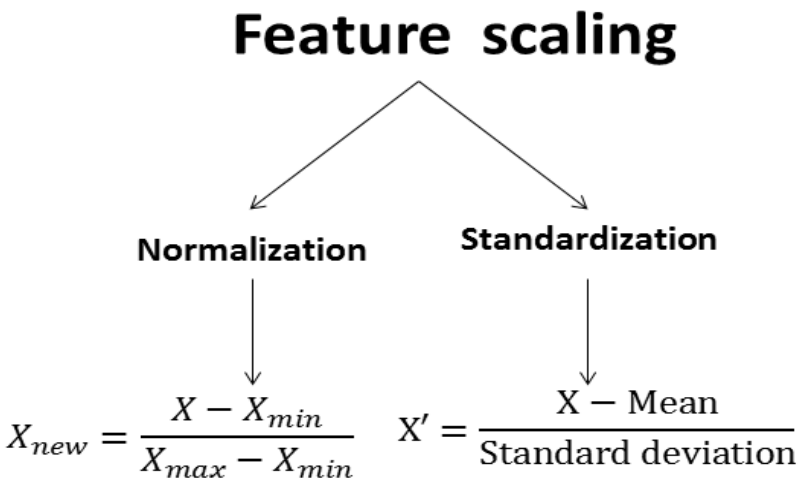
- Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range.
- In feature scaling, we put our variables in the same range and in the same scale so that no variable dominates the other variable.
- There are two ways to perform feature scaling in machine learning:-

Standardization and normalization are techniques used in data preprocessing to transform input data into a standard or normalized format.

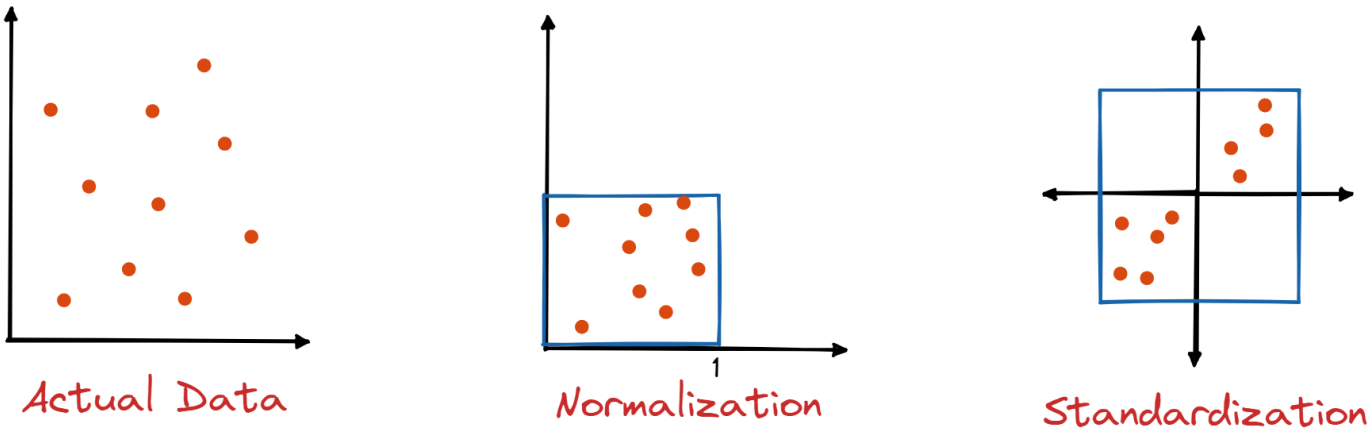
1. Standardization is the process of transforming data so that it has a mean of 0 and a standard deviation of 1. This is achieved by subtracting the mean of the data and then dividing by the standard deviation. Standardization helps to remove the scale of the data and makes it easier to compare different features or variables on the same scale. Standardization is particularly useful when dealing with features that have different scales or units.

2. Normalization is the process of scaling the values of a variable to a range between 0 and 1. This is achieved by subtracting the minimum value and then dividing by the range (i.e., the difference between the maximum and minimum values). Normalization helps to remove the effects of differences in the absolute magnitude of variables and brings all variables to the same scale. Normalization is particularly useful when dealing with features that have very different ranges or when the absolute values of the variables are not important.

Both standardization and normalization are commonly used in machine learning and data analysis to improve the performance of models, reduce bias, and improve the interpretability of results. The choice between standardization and normalization depends on the nature of the data and the requirements of the analysis or modeling task.



- We can use any of the feature scaling techniques that are standardization or normalization.
- After performing standardization the data shrinks in between [-1,1]
- And after performing normalization the data shrinks in between [0,1]

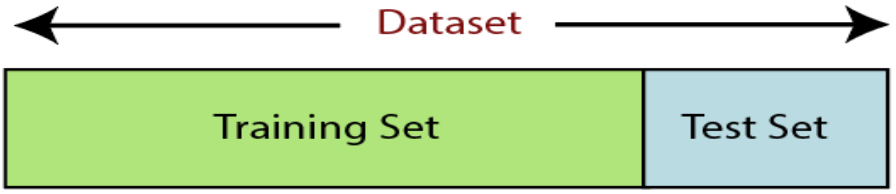


Splitting of dataset into training and testing

Splitting a dataset into training and testing sets is a common technique in machine learning to evaluate the performance of a model. The idea is to use a subset of the data to train the model, and another subset to test its performance. The training set is used to fit the model to the data, while the testing set is used to evaluate how well the model generalizes to new data.

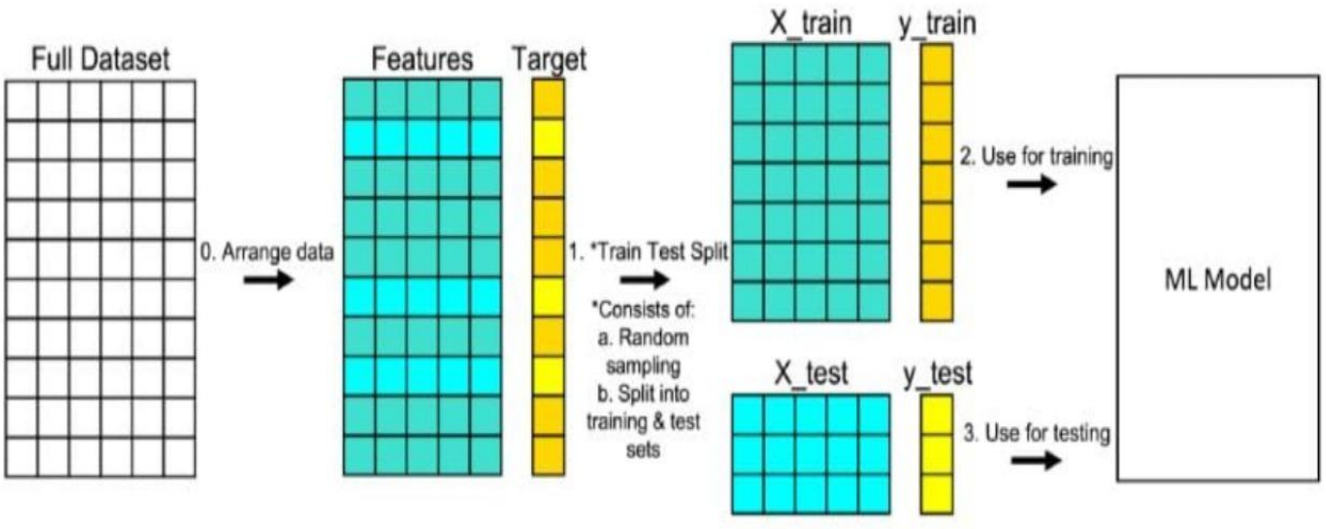
The usual approach is to randomly partition the dataset into two sets: a training set and a testing set. The ratio of the two sets can vary depending on the size of the dataset and the specific problem being solved.

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.



- **Training Set:** A subset of dataset to train the machine learning model, and we already know the output.
- **Test set:** A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.
- For splitting the dataset we use the below code:-

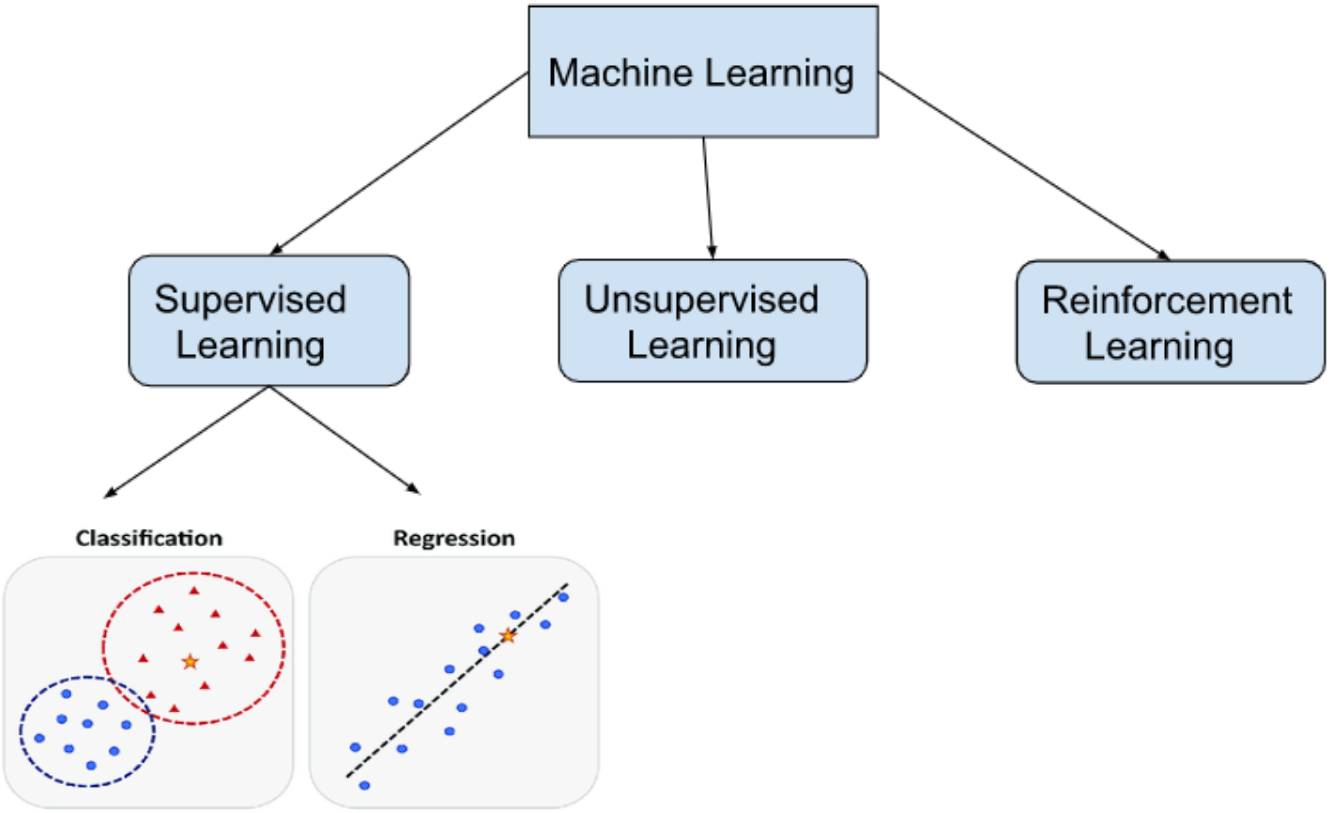
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```



Now we are going to train the model using the training dataset and apply the Machine learning algorithms.

5. MACHINE LEARNING ALGORITHMS USED IN OUR PROJECT:-

Classification of Machine Learning:-



Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that enables computers to learn from data without being explicitly programmed. The field of machine learning can be broadly divided into three categories: supervised learning, unsupervised learning, and reinforcement learning.

1. **Supervised Learning**: Supervised learning is a type of machine learning in which the model is trained on labeled data. The labeled data consists of input features and their corresponding output labels. The goal of supervised learning is to learn a function that maps input features to their respective output labels. Once the model is trained, it can be used to predict the output label for new input data. Supervised learning is used in a wide range of applications such as image classification, speech recognition, and sentiment analysis.

Supervised learning algorithms can be further categorized into two types: classification and regression.

In supervised learning, there are two main types of problems that can be addressed: classification and regression.

Classification is a type of supervised learning where the goal is to predict a categorical or discrete output variable. The input data is used to train a model that learns to map input features to discrete output labels. The output labels could represent different classes, such as whether an email is spam or not, or the type of flower based on its characteristics.

On the other hand, regression is a type of supervised learning where the goal is to predict a continuous output variable. The input data is used to train a model that learns to map input features to a continuous output value. The output value could represent a price, temperature, or any other real-valued quantity.

To summarize, the main difference between classification and regression is the type of output variable being predicted. Classification deals with discrete or categorical variables, while regression deals with continuous variables.

Unsupervised Learning: Unsupervised learning is a type of machine learning in which the model is trained on unlabeled data. The goal of unsupervised learning is to find patterns or structure in the data. Unlike supervised learning, there are no output labels provided during training. Unsupervised learning is used in a wide range of applications such as anomaly detection, clustering, and dimensionality reduction.

Reinforcement Learning: Reinforcement learning is a type of machine learning in which the model learns by interacting with an environment. The model receives feedback in the form of rewards or punishments for its actions. The goal of reinforcement learning is to learn a policy that maximizes the cumulative reward over time. Reinforcement learning is used in a wide range of applications such as game playing, robotics, and autonomous vehicles.

In conclusion, supervised, unsupervised, and reinforcement learning are the three main categories of machine learning. Each category has its own set of algorithms and applications. The choice of the appropriate category and algorithm depends on the nature of the data and the problem at hand. Understanding the differences between these categories is essential for anyone interested in applying machine learning to real-world problems.

CLASSIFICATION AND REGRESSION-

Classification and regression are two fundamental tasks in machine learning. Both tasks involve predicting a target variable based on input features, but they differ in the type of target variable being predicted.

Classification is a supervised learning task in which the target variable is a categorical variable. The goal is to learn a model that can predict the class of an input instance based on its features. For example, a classification model might be trained to predict whether an email is spam or not based on its content and metadata.

Regression is another supervised learning task in which the target variable is a continuous variable. The goal is to learn a model that can predict a numerical value based on input features. For example, a regression model might be trained to predict the price of a house based on its location, size, and other features.

In classification, the output of the model is a probability distribution over the possible classes, and the class with the highest probability is chosen as the predicted class. In regression, the output of the model is a single numerical value.

Both classification and regression can be implemented using various machine learning algorithms, including decision trees, neural networks, support vector machines, and others. The choice of algorithm depends on the specific task and the characteristics of the data.

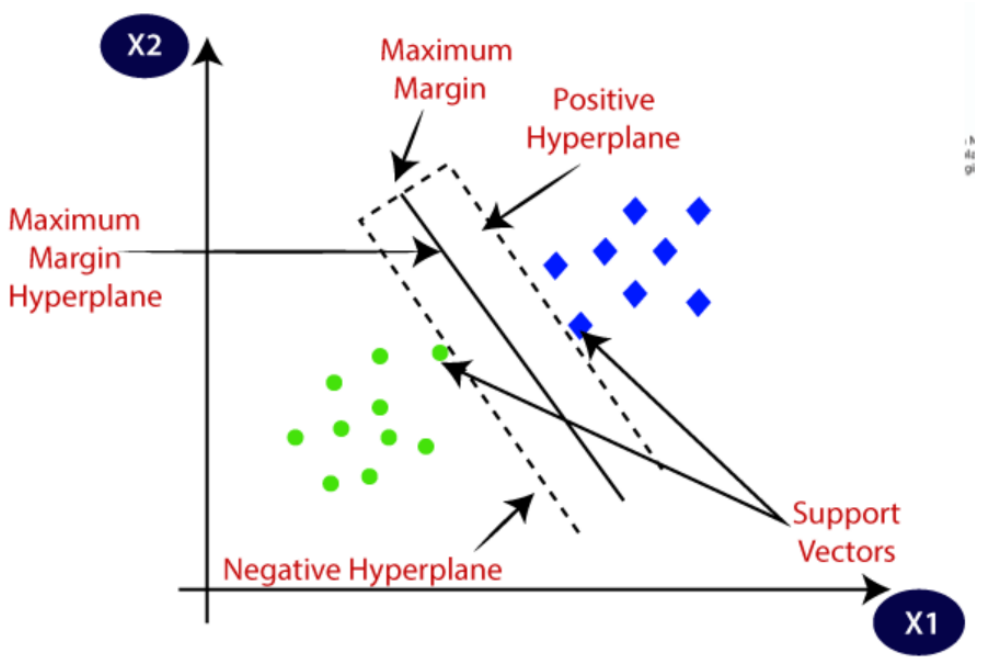
Overall, classification and regression are both important tools in machine learning that can be used to solve a wide range of problems. By leveraging these techniques, we can make predictions and gain insights from data to inform decision-making and drive progress in various fields.

ALGORITHMS USED

5.1 SUPPORT VECTOR MACHINE

Support vector machine (SVM) is a powerful and widely used algorithm in the field of machine learning. It is a type of supervised learning algorithm that can be used for both classification and regression tasks. The main objective of SVM is to find the hyperplane in a high-dimensional space that best separates the different classes. In this way, SVM can be used to classify new data points based on their features.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed a Support Vector Machine.



Followings are important concepts in SVM -

Support Vectors - Data Points that are closest to the hyperplane are called support vectors. Separating lines will be defined with the help of these data points.

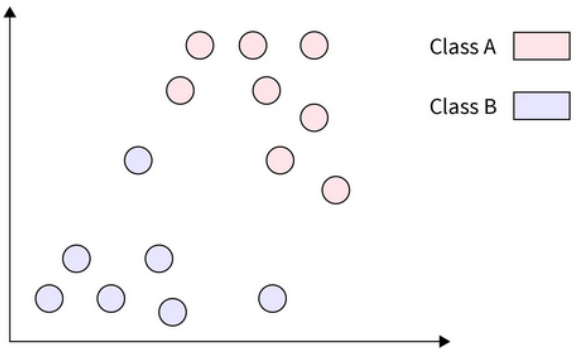
Hyperplane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closest data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. A large margin is considered a good margin and a small margin is considered a bad margin.

5.1.1 Linear SVM

Linear SVM is a popular variant of the Support Vector Machine (SVM) algorithm used for binary classification tasks. The main idea behind linear SVM is to find the optimal hyperplane that separates the two classes in a given dataset.

In linear SVM, the input data is not transformed into a higher-dimensional space, as is done in other types of SVM. Instead, the decision boundary is a linear function of the input features. The decision boundary is essentially a hyperplane that separates the two classes. The goal is to find the hyperplane that maximizes the margin between the two classes. The margin is defined as the perpendicular distance between the hyperplane and the closest data points of each class. The data points that lie closest to the hyperplane are called support vectors.

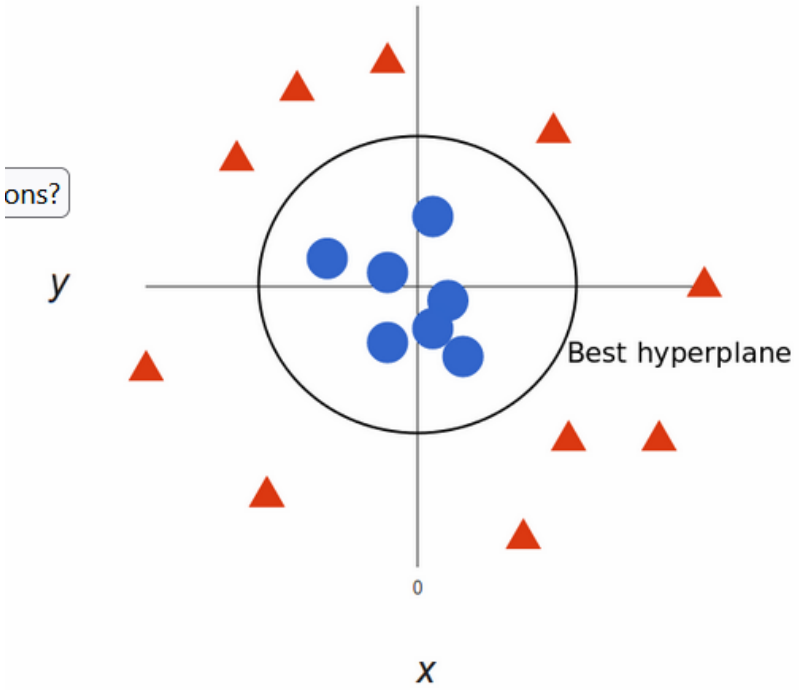


Linear SVM has several advantages. It is computationally efficient, and the decision boundary is simple and interpretable. It is also less prone to overfitting, making it a suitable choice for high-dimensional datasets

In summary, linear SVM is a powerful algorithm for binary classification tasks. It finds the optimal hyperplane that maximizes the margin between the two classes in the input space. The simplicity and efficiency of linear SVM make it a popular choice in various applications such as text classification, image classification, and bioinformatics

5.1.2 NON LINEAR SVM

Non-linear SVM is a variant of the Support Vector Machine (SVM) algorithm used for binary classification tasks where the decision boundary between the two classes cannot be represented by a linear hyperplane in the original input space. In non-linear SVM, the input data is transformed into a higher-dimensional space using a nonlinear function called a kernel function. The kernel function maps the input data into a higher-dimensional feature space where the data becomes separable by a linear hyperplane..



The choice of kernel function is crucial in non-linear SVM. The most commonly used kernel functions are the polynomial kernel, radial basis function (RBF) kernel, and sigmoid kernel. The polynomial kernel maps the data into a higher-dimensional space using a polynomial function. The RBF kernel maps the data into an infinite-dimensional space using a Gaussian function. The sigmoid kernel maps the data into a higher-dimensional space using a sigmoid function.

Non-linear SVM has several advantages. It can handle non-linearly separable data and can achieve high accuracy by selecting an appropriate kernel function. Moreover, non-linear SVM is less prone to overfitting compared to other non-linear algorithms. However, the choice of kernel function and the hyperparameters requires careful tuning, which can be computationally expensive.

In summary, non-linear SVM is a powerful algorithm for binary classification tasks where the decision boundary between the two classes cannot be represented by a linear hyperplane. Non-linear SVM maps the input data into a higher-dimensional space using a kernel function, making the data separable by a linear hyperplane. The choice of kernel function and hyperparameters is critical in non-linear SVM and requires careful tuning for obtaining good performance.

5.2 NAIVE BAYES

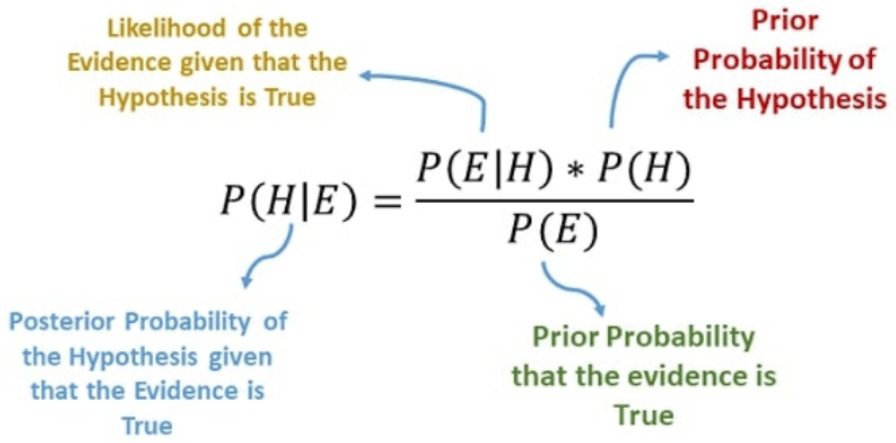
Naive Bayes is a probabilistic machine learning algorithm used for classification tasks. It is based on Bayes' theorem and the assumption of conditional independence between the features. The algorithm is simple, fast, and effective in many real-world applications.

The Naive Bayes algorithm assumes that each feature in the dataset is independent of all other features, given the class label. This assumption is known as the "naive" assumption, which simplifies the computation of the probability distribution of the features. The algorithm estimates the probability of each class label given the input features and selects the class label with the highest probability.

The Naive Bayes algorithm can be formulated as follows:

P(class | features) = P(features | class) * P(class) / P(features)

where P(class | features) is the probability of the class label given the input features, P(features | class) is the conditional probability of the features given the class label, P(class) is the prior probability of the class label, and P(features) is the probability of the input features.



The algorithm estimates the conditional probability of the features given the class label, P(features | class), using the training data. This can be done using various probability distributions such as Gaussian, Bernoulli, and Multinomial. The choice of probability distribution depends on the nature of the input features. For example, Gaussian distribution is used for continuous features, while Bernoulli distribution is used for binary features.

The algorithm estimates the prior probability of the class label, P(class), using the training data. This can be done by counting the number of occurrences of each class label in the training data and dividing by the total number of training samples.

The algorithm estimates the probability of the input features, P(features), using the training data. This can be done by assuming that the input features are independent and computing the product of the conditional probabilities of each feature given the class label.

The Naive Bayes algorithm can handle high-dimensional data and can be trained on small datasets. Moreover, it is less prone to overfitting compared to other algorithms, making it suitable for classification tasks with limited data.

However, the Naive Bayes algorithm may not work well when the naive assumption of conditional independence between the features is violated. Moreover, the algorithm may produce unreliable probability estimates when the number of training samples is small, and the dataset is imbalanced.

In summary, Naive Bayes is a simple, fast, and effective algorithm for classification tasks based on Bayes' theorem and the assumption of conditional independence between the features. The algorithm estimates the probability of each class label given the input features and selects the class label with the highest probability. The Naive Bayes algorithm is widely used in various applications such as text classification, spam filtering, and sentiment analysis.

Types of Naive Bayes Classifiers

There are three main types of Naive Bayes classifiers: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. Each type of classifier assumes a different probability distribution of the input features and is suitable for different types of data.

5.2.1 Gaussian Naive Bayes

Gaussian Naive Bayes is a variant of the Naive Bayes algorithm that assumes a Gaussian (normal) distribution of the input features. It is a simple and fast algorithm that is particularly useful for continuous input features that can be modeled as real numbers. Gaussian Naive Bayes is widely used in various applications such as text classification, image classification, and medical diagnosis.

The Gaussian Naive Bayes algorithm assumes that the probability distribution of each input feature for each class is Gaussian.

The probability density function (PDF) of a Gaussian distribution is given by:

$$f(x \mid \mu, \sigma) = (1 / \text{sqrt}(2\pi\sigma^2)) * \exp(-((x - \mu)^2 / 2\sigma^2))$$

where x is the input feature value, μ is the mean of the feature for a specific class, σ is the standard deviation of the feature for that class, and exp is the exponential function.

The algorithm estimates the mean and standard deviation of each input feature for each class using the training data. Then, it computes the probability density function of each input feature for each class using the above equation. Finally, it computes the joint probability of all input features for each class using the product rule and estimates the conditional probability of each class given the input features using Bayes' theorem.

The Gaussian Naive Bayes algorithm can be formulated as follows:

1. Estimate the mean and standard deviation of each input feature for each class using the training data.
2. For each input feature and class, compute the probability density function of the feature using the Gaussian distribution.
3. For each class, compute the joint probability of all input features

In summary, Gaussian Naive Bayes is a variant of the Naive Bayes algorithm that assumes a Gaussian distribution of the input features. It is a simple and fast algorithm that is particularly useful for continuous input features that can be modeled as real numbers. The algorithm estimates the mean and standard deviation of each input feature for each class using the training data and computes the probability density function of each input feature for each class using the Gaussian distribution. Finally, it estimates the conditional probability of each class given the input features using Bayes' theorem.

5.2.2 Bernoulli Naive Bayes

Bernoulli Naive Bayes is a variant of the Naive Bayes algorithm that is used for binary classification problems where the input features are binary variables (e.g., 0 or 1). It is particularly useful for text classification tasks where the input features represent the presence or absence of certain words in a document.

The Bernoulli Naive Bayes algorithm assumes that each input feature is binary and follows a Bernoulli distribution. The Bernoulli distribution is a discrete probability distribution that takes the value 1 with probability p and the value 0 with probability 1-p, where p is the probability of success.

The Bernoulli Naive Bayes algorithm can be formulated as follows:

1. For each input feature and class, compute the probability of the feature given the class using the training data. T
2. For each input feature and class, compute the probability of the feature not given the class
3. Compute the prior probability of each class using the training data:
4. Compute the joint probability of all input features for each class
5. Compute the probability of the input features using the joint probability of all features for each class and the prior probability of each class:
6. Compute the conditional probability of each class given the input features using Bayes' theorem.

In summary, Bernoulli Naive Bayes is a variant of the Naive Bayes algorithm that is used for binary classification problems where the input features are binary variables. The algorithm assumes that each

input feature is binary and follows a Bernoulli distribution. The algorithm estimates the probability of each feature given the class using the training data and computes the joint probability of all input features for each class using the product rule. Finally, it estimates the conditional probability of each class given the input features using Bayes' theorem.

5.2.3 Multinomial Naive Bayes

Multinomial Naive Bayes is a variant of the Naive Bayes algorithm that is used for text classification problems where the input features are discrete counts of the number of times each word appears in a document. It is particularly useful for text classification tasks where the input features represent the frequency of occurrence of certain words in a document.

The Multinomial Naive Bayes algorithm assumes that each input feature follows a Multinomial distribution. The Multinomial distribution is a discrete probability distribution that represents the probability of observing a certain count of events in a fixed number of trials, where the events can occur with different probabilities.

The Multinomial Naive Bayes algorithm can be formulated as follows:

1. For each input feature and class, compute the probability of the feature given the class using the training data. This probability is estimated as the proportion of the total count of the feature in the class.
3. Compute the joint probability of all input features for each class using the product rule.
4. Compute the probability of the input features using the joint probability of all features for each class and the prior probability of each class.
5. Compute the conditional probability of each class given the input features using Bayes' theorem:

In summary, Multinomial Naive Bayes is a variant of the Naive Bayes algorithm that is used for text classification problems where the input features are discrete counts of the number of times each word appears in a document. The algorithm assumes that each input feature follows a Multinomial distribution. The algorithm estimates the probability of each feature given the class using the training data and computes the joint probability of all input features for each class using the product rule. Finally, it estimates the conditional probability of each class given the input features using Bayes' theorem.

6 Model Accuracy and Testing

Now as we have trained our model using machine learning algorithms, we are going to test our model using the test dataset which we split earlier from the original dataset. Now the model will predict the output which is whether the patient has heart disease or not.

Compare the predictions to the actual labels (i.e., whether the patient has heart disease or not) in the test set to evaluate the model's performance.

Common metrics for classification problems include confusion, accuracy, precision, recall, and F1 score.

Now let us calculate the accuracy of our trained model There are several ways to calculate the accuracy of a trained machine learning model for predicting heart disease, but one of the most common methods is to use the following formula:

Accuracy = (number of correct predictions) / (total number of predictions)

To calculate accuracy, you can compare the model's predictions to the actual labels (i.e., whether the patient has heart disease or not) in the test set.

For example,

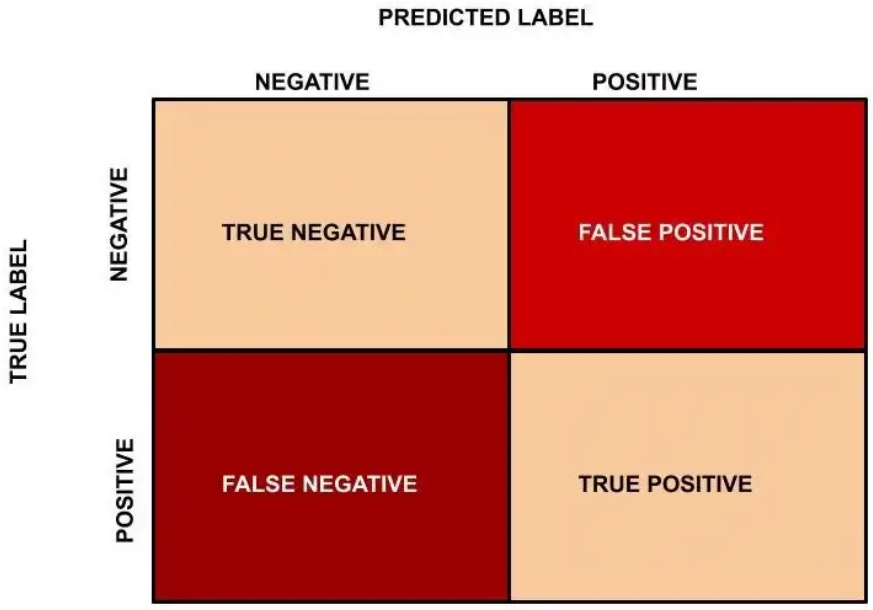
- if the model correctly predicts that a patient has heart disease and the patient actually has heart disease, that is considered a **true positive**.

- If the model correctly predicts that a patient does not have heart disease and the patient actually does not have heart disease, that is considered a **true negative**.

- If the model incorrectly predicts that a patient has heart disease and the patient actually does not have heart disease, that is considered a **false positive.**

- If the model incorrectly predicts that a patient does not have heart disease and the patient actually has heart disease, that is considered a **false negative.**

Now we will introduce the ***confusion matrix*** which is required to compute the ***accuracy*** of the machine learning algorithm in classifying the data into its corresponding labels



The formula for accuracy can be expressed in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) as follows:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

We are going to use libraries such as scikit-learn, which have inbuilt functions to calculate accuracy, precision, recall, and f1-score.

It's important to note that accuracy is not the only metric that should be considered when evaluating a machine-learning model. Other important metrics include precision, recall, and F1 score. These metrics provide a more complete picture of the model's performance, especially when dealing with imbalanced datasets.

Precision-:

Precision is a metric used to evaluate the performance of a machine learning model, specifically in classification problems. It is a measure of the number of true positive predictions made by the model out of the total number of positive predictions made by the model.

Formally, precision is defined as

Precision = True Positives / (True Positives + False Positives)

where:

True Positives (TP) are the number of times the model correctly predicts that a patient has heart disease.

False Positives (FP) are the number of times the model incorrectly predicts that a patient has heart disease when the patient actually does not have heart disease.

In other words, precision tells us how often the model is correct when it predicts that a patient has heart disease. High precision means that the model has a low rate of false positives, which is desirable in many cases, such as medical diagnosis.

Recall-:

Recall is also a metric used to evaluate the performance of a machine learning model. It is a measure of the number of true positive predictions made by the model out of the total number of actual positive cases.

Formally, recall is defined as

Recall = True Positives / (True Positives + False Negatives)

where:

False Negatives (FN) are the number of times the model incorrectly predicts that a patient does not have heart disease when the patient actually has heart disease.

In other words, recall tells us how many of the actual positive cases the model is able to identify. High recall means that the model has a low rate of false negatives, which is desirable in many cases, such as medical diagnoses.

F1 score-:

The F1 score is also a metric used to evaluate the performance of a machine learning model. It is a measure that combines precision and recall, and it is calculated as the harmonic mean of precision and recall.

Formally, the F1 score is defined as

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

The F1 score ranges from 0 to 1, where a score of 1 indicates perfect precision and recall, and a score of 0 indicates that the model has not made any true positive predictions.

7. RESULT

After performing the machine learning approach for training and testing we find that accuracy of the SVM is better compared to Naive Bayes. Accuracy is calculated with the support of the confusion matrix of each algorithm, here the number count of TP, TN, FP, FN is given and using the equation of accuracy, value has been calculated and it is concluded that extreme gradient boosting is best with 91% accuracy and the comparison is shown below.

TABLE: Accuracy comparison of algorithms

Algorithm	Accuracy
Support vector machines	91%
Naive Bayes(Gaussian)	87.72%
Naive Bayes(Bernoulli)	85.96%

8. CODING

1 DATA PREPROCESSING

- Firstly we have loaded the dataset-

```
In [2]: dataset=pd.read_csv(r'C:\Users\shamb\Documents\Project_MachineLearning\main.csv')
```

- Description of Dataset

Description

```
In [6]: dataset.describe()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.00
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.75
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.03
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.00
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.00
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.00
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.00
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.00

- Checking for Null Values

```
In [7]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

we have no missing values

This code will print the number of null values in each column of the dataframe. If there are any null values in the dataset, you can then decide on a strategy to handle them, such as imputation or deletion of the affected rows or columns.

- **Checking for Duplicate Values**

We use the duplicated function to find rows that are duplicates of previous rows in the DataFrame. Finally, we print the number of duplicate rows found.

For removing the duplicate rows from the DataFrame, we have used the drop_duplicates function

Are there duplicate value present

```
In [8]: dataset.duplicated().sum()
Out[8]: 723

In [9]: # 723 rows were duplicate so to remove the duplicate rows use drop duplicate function
        dataset=dataset.drop_duplicates()

In [10]: dataset.shape
Out[10]: (302, 14)
```

Analysing the 'target' variable

```
In [12]: dataset["target"].describe()
Out[12]: count    302.000000
         mean       0.543046
         std        0.498970
         min        0.000000
         25%        0.000000
         50%        1.000000
         75%        1.000000
         max        1.000000
         Name: target, dtype: float64

In [13]: dataset["target"].unique()
Out[13]: array([0, 1], dtype=int64)
```

Clearly, this is a classification problem, with the target variable having values '0' and '1'

Checking correlation between columns

In [14]:
```
print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

```
target      1.000000
exang       0.435601
cp          0.432080
oldpeak     0.429146
thalach     0.419955
ca          0.408992
slope       0.343940
thal        0.343101
sex         0.283609
age         0.221476
trestbps    0.146269
restecg     0.134874
chol        0.081437
fbs         0.026826
Name: target, dtype: float64
```

#This shows that most columns are moderately correlated with target, but 'fbs' is very weakly correlated.

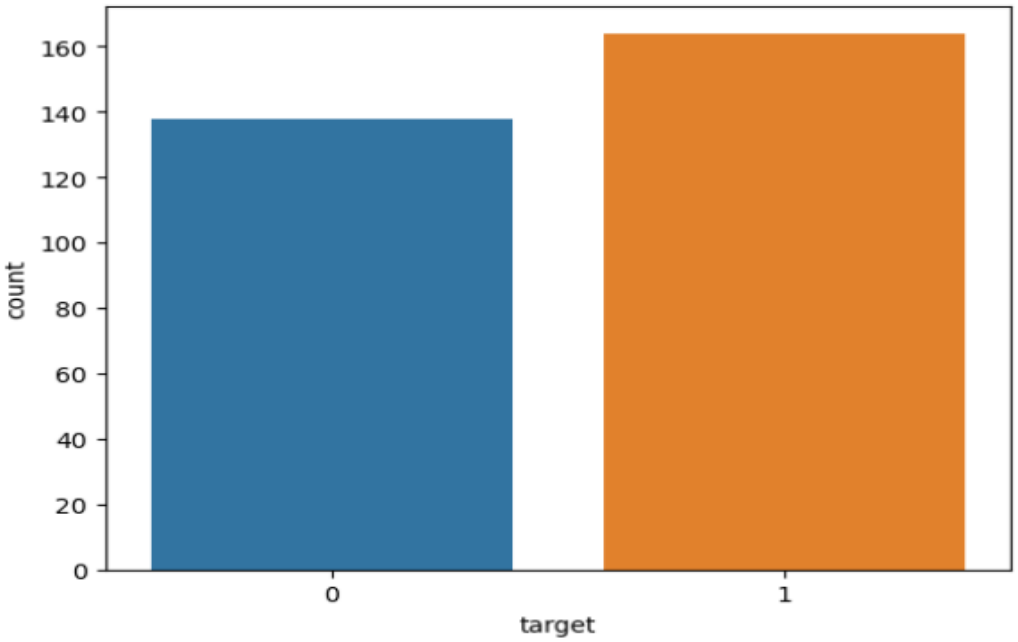
```
y = dataset["target"]

sns.countplot(y)


target_temp = dataset.target.value_counts()

print(target_temp)
```

```
1    164
0    138
Name: target, dtype: int64
```



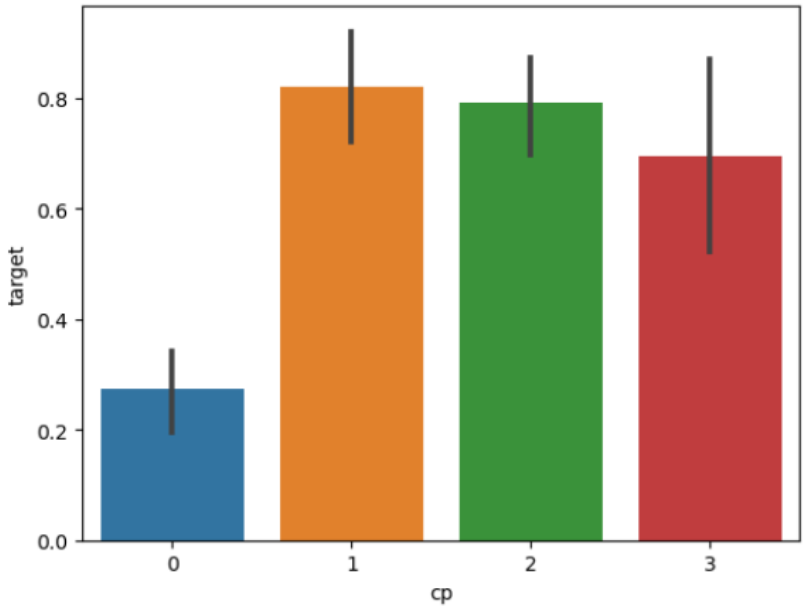
```
: print("Percentage of patience with heart problems: "+str(y.where(y==1).count()*100/303))
  print("Percentage of patience with heart problems: "+str(y.where(y==0).count()*100/303))
```

```
Percentage of patience with heart problems: 54.12541254125413
Percentage of patience with heart problems: 45.54455445544554
```

Analyzing the 'Chest Pain Type' feature

```
In [19]: dataset["cp"].unique()
Out[19]: array([0, 1, 2, 3], dtype=int64)

In [20]: sns.barplot(dataset["cp"],y)
Out[20]: <AxesSubplot:xlabel='cp', ylabel='target'>
```



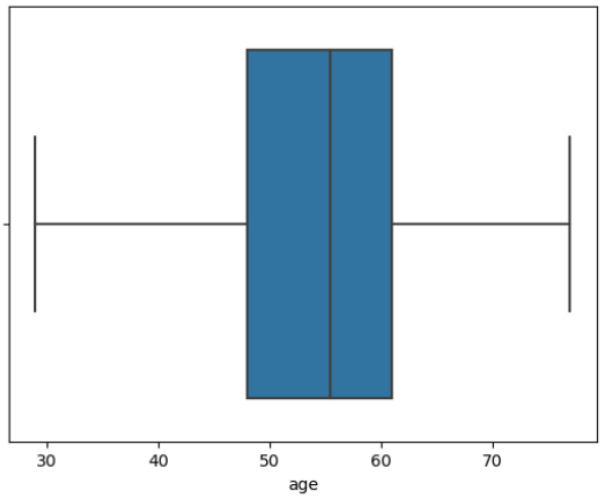
We notice, that chest pain of '0', i.e. the ones with typical angina are much less likely to have heart problems

Similar analysis is done for the rest of the features also.

Outliers

AGE

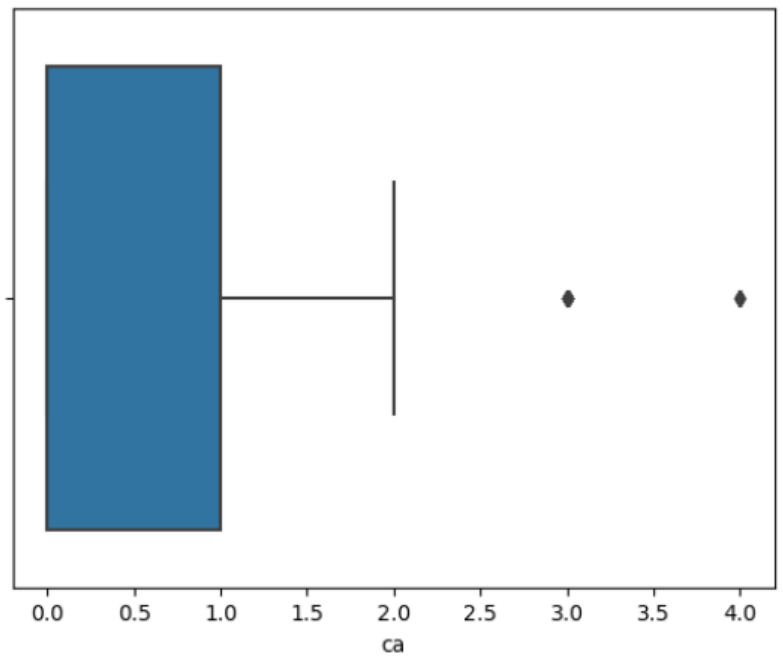
```
In [37]: sns.boxplot(dataset['age'])
Out[37]: <AxesSubplot:xlabel='age'>
```



No outliers present in AGE

Ca

```
In [39]: sns.boxplot(dataset['ca'])
Out[39]: <AxesSubplot:xlabel='ca'>
```



so here in the bar plot we can see two black dots that represent the outliers.

Removing Outliers

outliers in trestbps

In [45]:
```
# Finding the IQR
percentile25 = dataset['trestbps'].quantile(0.25)
percentile75 = dataset['trestbps'].quantile(0.75)
iqr = percentile75 - percentile25
iqr
```

Out[45]: 20.0

In [46]:
```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
print("Upper limit",upper_limit)
print("Lower limit",lower_limit)
```

```
Upper limit 170.0
Lower limit 90.0
```

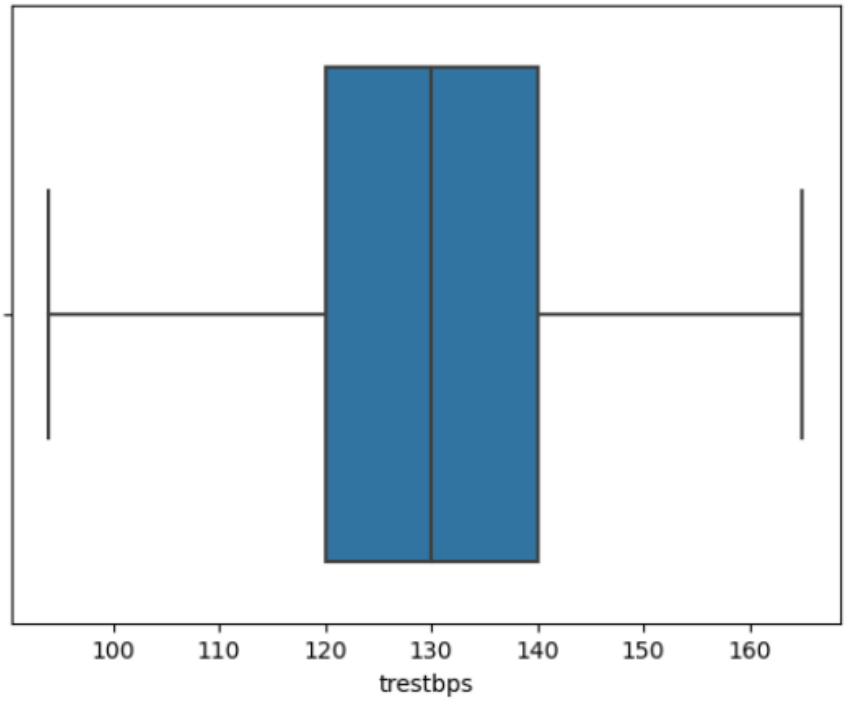
In [47]:
```
dataset[dataset['trestbps'] > upper_limit]
```

Out[47]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
29	55	0	0	180	327	0	2	117	1	3.4	1	0	2	0
47	66	0	0	178	228	1	1	165	1	1.0	1	2	3	0
87	59	0	0	174	249	0	1	143	1	0.0	1	0	2	0
137	64	0	0	180	325	0	1	154	1	0.0	2	0	2	1
151	54	1	1	192	283	0	0	195	0	0.0	2	1	3	0
175	56	0	0	200	288	1	0	133	1	4.0	0	2	3	0
343	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
396	68	1	2	180	274	1	0	150	1	1.6	1	0	3	0
528	59	1	3	178	270	0	0	145	0	4.2	0	0	3	1

In [49]:
```
dataset = dataset[dataset['trestbps'] < upper_limit]
dataset
```

```
In [50]: sns.boxplot(dataset['trestbps'])
Out[50]: <AxesSubplot:xlabel='trestbps'>
```



Splitting of dataset into x and y:-

```
In [11]: features=[ 'age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca',
         X=ps.loc[:, features]
         y=ps.loc[:,['target']]
```

Splitting of dataset into test and train

```
In [66]: y=dataset['target']
         x=dataset.drop(['target'],axis=1)

In [67]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=17)

         X_train.shape, X_test.shape
Out[67]: ((226, 13), (57, 13))
```

Normalization using min max scaler

MinMax Scaler

```
In [68]: from sklearn.preprocessing import MinMaxScaler
         scaler=MinMaxScaler()
         preprocessed_data=scaler.fit_transform(x)
         df=pd.DataFrame(preprocessed_data)
```

Training using SVM

SVM

```
In [69]: #Create a svm Classifier
         from sklearn import svm
         ml = svm.SVC(kernel='linear') # Linear Kernel

         #Train the model using the training sets
         ml.fit(X_train, y_train)

         #Predict the response for test dataset
         y_pred = ml.predict(X_test)
```

```
In [70]: print("accuracy of train is : ",ml.score(X_train,y_train)*100)

         accuracy of train is :  85.39823008849558
```

```
In [71]: print("accuracy of test is : ",ml.score(X_test,y_test)*100)

         accuracy of test is :  91.22807017543859
```

Gaussian Naive Bayes

```
In [72]: from sklearn.naive_bayes import GaussianNB

         nb = GaussianNB()

         nb.fit(X_train,y_train)

         Y_pred_nb = nb.predict(X_test)
```

```
In [73]: Y_pred_nb.shape

Out[73]: (57,)
```

```
In [74]: from sklearn.metrics import accuracy_score
         score_nb = round(accuracy_score(Y_pred_nb,y_test)*100,2)
         print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")

         The accuracy score achieved using Naive Bayes is: 87.72 %
```

Training using Bernoulli Naive Bayes

Bernoulli Naive Bayes

In [77]:
```
from sklearn.naive_bayes import BernoulliNB

nb = BernoulliNB()

nb.fit(X_train,y_train)

Y_pred_nb = nb.predict(X_test)
```

In [78]:
```
Y_pred_nb.shape
```

Out[78]: (57,)

In [79]:
```
from sklearn.metrics import accuracy_score
score_nb = round(accuracy_score(Y_pred_nb,y_test)*100,2)
print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)+" %")
```

```
The accuracy score achieved using Naive Bayes is: 85.96 %
```

:
```
from sklearn.metrics import classification_report
cr = classification_report(y_pred,y_test)

print("Accuracy is:")
print(cm)
```

```
Accuracy is:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88        19
           1       1.00      0.87      0.93        38

    accuracy                           0.91        57
   macro avg       0.90      0.93      0.91        57
weighted avg       0.93      0.91      0.91        57
```

Coding for designing website : -

app.py

```python
from flask import Flask,render_template, request , redirect,url_for
import numpy as np
import pickle
import pandas as pd

model = pickle.load(open('heart.pkl', 'rb'))

app = Flask(__name__)

@app.route('/')
def start():
    return render_template('index.html')

@app.route('/index.html')
def home():
    return render_template('index.html')

@app.route('/about.html')
def about():
    return render_template('about.html')

@app.route('/heartform.html')
def form():
    return render_template('heartform.html')

@app.route('/submit', methods=['POST'])
def predict():
    age = request.form['age']
    sex = request.form['sex']
```

app.py

app.py > start

```
    age = request.form['age']
    sex = request.form['sex']
    cp = request.form['cp']
    trestbps = request.form['trestbps']
    chol = request.form['chol']
    fbs = request.form['fbs']
    restecg = request.form['restecg']
    thalach = request.form['thalach']
    exang = request.form['exang']
    oldpeak = float(request.form['oldpeak'])
    slope = request.form['slope']
    ca = request.form['ca']
    thal = request.form['thal']

    # create a list of input features
    input_data = [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]
    #return(input_data)
     # changing the input_data to numpy array
    arr = np.array(input_data)
    # reshape the array as we are predicting for one instance
    input_data_reshaped = arr.reshape(1,-1)

    prediction = model.predict(input_data_reshaped)


    # return the prediction result
    return render_template('next.html',data=prediction)
```

9. USER INTERFACES:-

The user interface (UI) design for this heart disease prediction system would be a web-based application that includes several key components. Some of these components include :

1. Navigation menu: A clear and easy-to-use navigation menu that allows users to access different sections of the UI, such as the home page and about.

2. Input forms: Forms that guide users through the process of inputting patient data, such as demographic information, medical history, and lab results. These forms are designed to be easy to understand and fill out, with clear instructions and error messages.

3. Results display: A clear and easy-to-understand display of the predictions that the system generates the chances of having heart disease.

Description of the User Interface

The user interface (UI) for our heart disease prediction system would be designed to be intuitive and user-friendly. The main goal of the UI design is to create an environment where users can easily input data and receive clear and accurate predictions. To achieve this, the UI would be developed with a clean and modern design that has a clear hierarchy of information and an easy-to-use navigation menu.

1. Navigation Menu: The navigation menu would be located at the top of the UI and would include options to access the home page, about page, and a button to check your heart. The home page would be the default landing page and would provide users with a brief overview of the system's purpose and features.
2. About Page: The about page would provide users with a brief introduction to the system, its purpose, and how to use it. The page would also provide instructions on how to fill out the input form, the type of data that should be inputted, and how to interpret the results.
3. Input Form: The check your heart page would consist of a series of forms that guide users through the process of inputting patient data. The forms would be designed to be easy to understand and fill out, with clear instructions and error messages. Users would be able to input data such as demographics, medical history, lab results, and other relevant information.
4. The input form would have a user-friendly design and clear labeling for all input fields. Users would be provided with clear instructions on how to input their data and what each field represents. The input form would be designed to allow users to easily navigate between input fields and to provide a seamless input experience.
5. Data Processing and Prediction: After inputting the data, the system would process the data and generate predictions. The predictions would be displayed in a clear and easy-to-understand format. The results would be presented to the user in a separate page to avoid confusion with the input form.

Overall, the UI would be designed to make it as easy as possible for users to input data and understand the results. It would have a modern and user-friendly design that guides users through the process of inputting data and interpreting the results. With a clear hierarchy of information and easy navigation, users would be able to quickly and efficiently access the information they need.

Tech stacks used in designing website and the backend connectivity -

FRONTEND - HTML,CSS

CONNECTIVITY - FLASK

1. HTML and CSS: HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides a structure for web content, such as text, images, and other multimedia. CSS (Cascading Style Sheets) is used to style HTML pages, making them look visually appealing and easy to navigate. Together, HTML and CSS provide the foundation for creating a website's layout and design.
2. Flask: Flask is a Python web framework used for building web applications. It provides tools for handling requests and responses, routing, and rendering templates. Flask is easy to use and flexible, making it an excellent choice for building web applications.
3. Machine Learning Model: Machine learning is a type of artificial intelligence that allows computers to learn from data and make predictions. A machine learning model is trained on a dataset to make predictions on new data. In the case of a heart disease prediction system, the model can be trained on a dataset of patient data that includes features such as age, gender, blood pressure, and cholesterol levels. The model can then predict the likelihood of a patient having heart disease based on these features.
4. Connecting Flask with Machine Learning Model: To connect Flask with a machine learning model, you need to use Flask's request and response objects to collect user input and display the prediction results. The Flask application can accept user input from an HTML form and pass it to the machine learning model. The model can then make a prediction and return the result to the Flask application, which can display it on the web page.
5. Deployment: Once the Flask application is built, it needs to be deployed on a web server to make it accessible to users. There are many options for deploying Flask applications, such as using a cloud-based service like Heroku or deploying it on a virtual private server (VPS).

In conclusion, the heart disease prediction system is a great application of machine learning and web development. By using HTML, CSS, Flask, and a machine learning model, you have created a user-friendly and effective tool for predicting the likelihood of heart disease. The deployment process is crucial to make the system available to users, and there are many options for hosting Flask applications.

Explanation of backend connectivity:-

Flask is a Python-based web framework that allows you to build web applications easily and quickly. Flask is a lightweight and flexible framework that provides an efficient and powerful toolset for building web applications. Flask provides several features, such as support for URL routing, templates, forms, and database connectivity. Flask is an open-source project, which means that the source code is freely available and can be modified as per the requirements of the project.

Overview of the Heart Disease Prediction Application The heart disease prediction application is a web application that utilizes a pre-trained machine learning model to predict the occurrence of heart disease based on the input data provided by the user. The application is built using the Flask web framework, which provides a robust and efficient platform for building web applications.

The application has several components that work together to provide the user with a seamless experience. The main components of the application are the Flask web server, the pre-trained machine learning model, and the HTML templates.

Flask Web Server The Flask web server is the core component of the heart disease prediction application. It is responsible for handling incoming requests from the client, processing the requests, and returning the appropriate responses. The Flask web server is built on top of the Werkzeug WSGI toolkit and the Jinja2 template engine.

The Flask web server is defined in the application.py file, which is the entry point for the application. The Flask server is created by instantiating a Flask object and specifying the name of the application. The Flask object is then used to define the various routes and endpoints of the application.

Routes and Endpoints The routes and endpoints of the heart disease prediction application are defined using the @app.route() decorator in the application.py file. The routes and endpoints define the various URLs that the user can access and the actions that can be performed on those URLs.

The main route of the application is the index route, which is defined using the @app.route('/') decorator. The index route is responsible for rendering the landing page of the application. The landing page provides the user with an overview of the application and allows them to navigate to the other pages of the application.

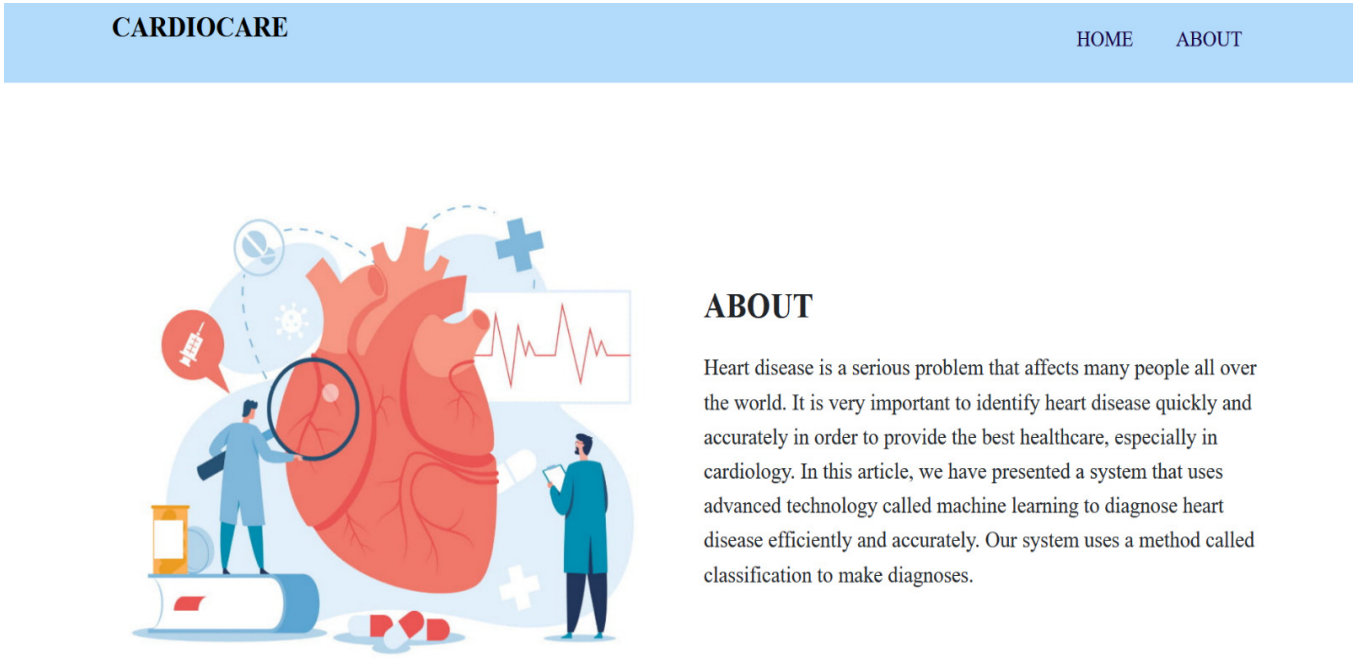
The other routes of the application are defined using the @app.route() decorator and correspond to different pages of the application. For example, the about route, defined using the @app.route('/about'), is responsible for rendering the about page of the application, which provides information about the project. Similarly, the heartform route, defined using the @app.route('/heartform'), is responsible for rendering the heartform page of the application, which displays a form for the user to enter the input data.

Templates The HTML templates of the heart disease prediction application are responsible for rendering the various pages of the application. The HTML templates are defined in the templates folder of the application and are written using the Jinja2 template engine.

The templates folder contains several HTML template files, including index.html, about.html, heartform.html, and next.html. The index.html template file is the landing page of the application and provides the user with an overview of the project. The about.html template file provides information about the project, while the heartform.html template file displays a form for the user to enter the input data. The next.html template file displays the predicted result of the machine learning model.

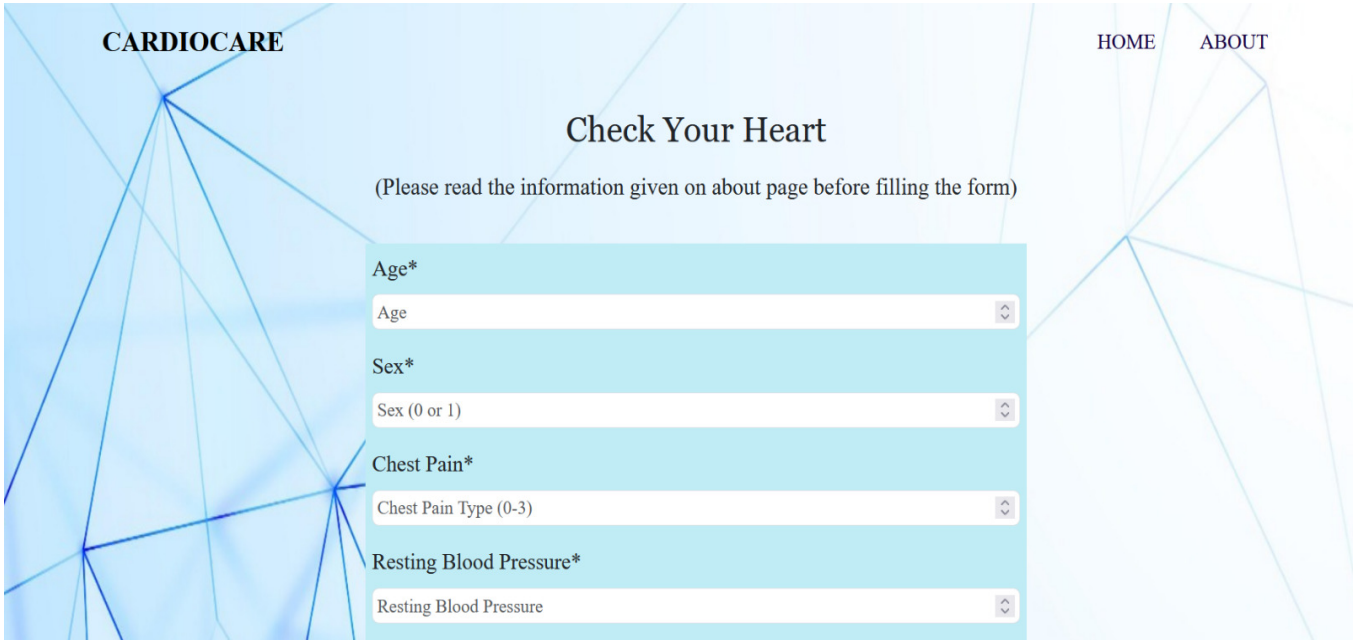
The predict() function is called when the user submits the form. The predict() function extracts the user input data from the form and passes it to the pre-trained machine learning model (heart.pkl) for prediction. The predicted result is then displayed on the next.html template file.

Some of the pictures of the website :-

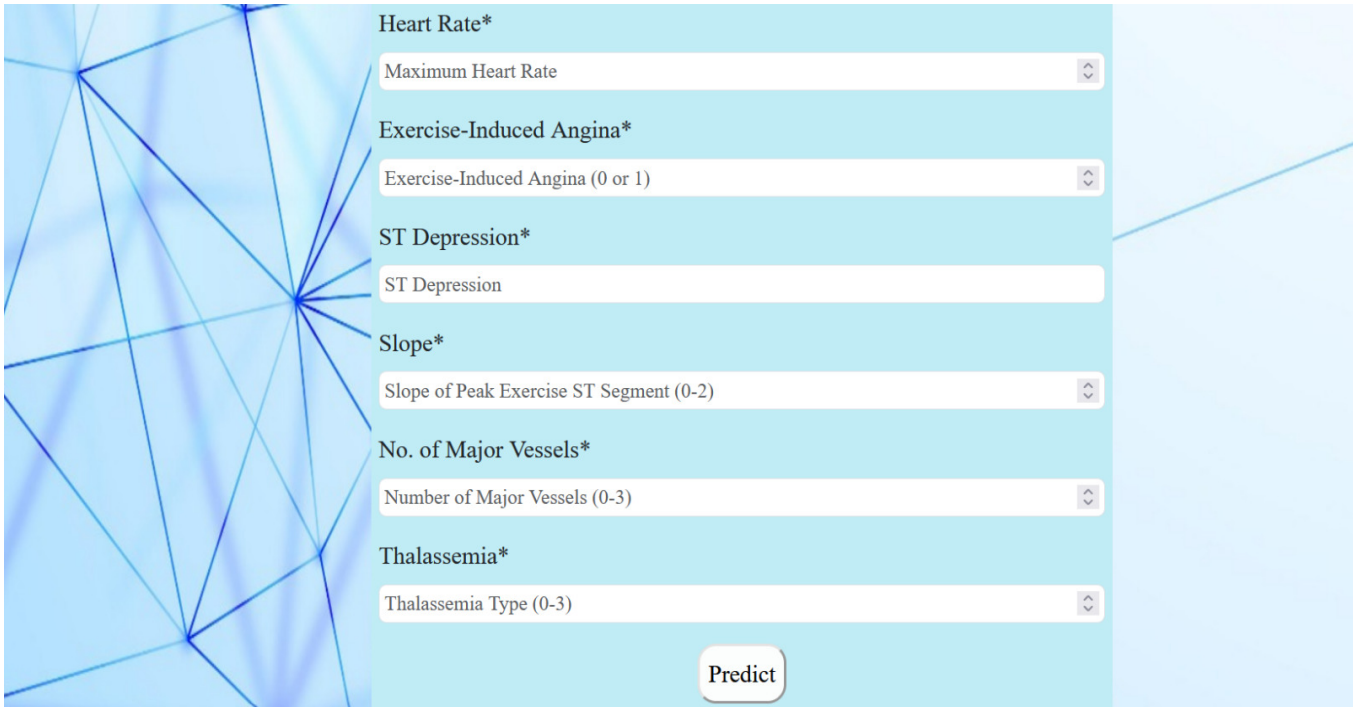


ABOUT

Heart disease is a serious problem that affects many people all over the world. It is very important to identify heart disease quickly and accurately in order to provide the best healthcare, especially in cardiology. In this article, we have presented a system that uses advanced technology called machine learning to diagnose heart disease efficiently and accurately. Our system uses a method called classification to make diagnoses.

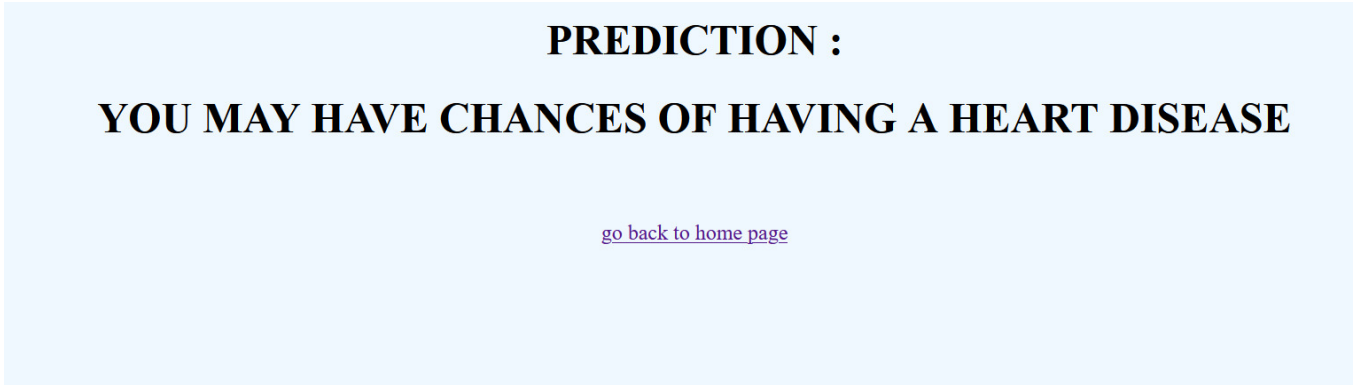

CARDIOCARE
HOME
ABOUT
Check Your Heart
(Please read the information given on about page before filling the form)
Age*
Age
Sex*
Sex (0 or 1)
Chest Pain*
Chest Pain Type (0-3)
Resting Blood Pressure*
Resting Blood Pressure


Cholesterol*
Serum Cholesterol
Fasting Blood Sugar*
Fasting Blood Sugar (0 or 1)
Resting ECG*
Resting ECG Results (0-2)
Heart Rate*
Maximum Heart Rate
Exercise-Induced Angina*
Exercise-Induced Angina (0 or 1)
ST Depression*
ST Depression
Slope*



After the user fills all the information and clicks on the predict button then a new web page appears showing the result whether the person is having heart disease or not.

eg.



PREDICTION :

YOU MAY NOT HAVE CHANCES OF HAVING A HEART DISEASE

go back to home page

10. CONCLUSION:-

In conclusion, a heart disease prediction system can be a valuable tool for healthcare professionals and individuals alike. By analyzing and processing relevant medical data, the system can provide accurate predictions and insights about an individual's risk of developing heart disease.

The development and implementation of such a system can have a significant impact on the prevention, early detection, and treatment of heart disease, which is one of the leading causes of death worldwide. With early identification of risk factors, healthcare providers can intervene early and implement preventive measures to reduce the risk of heart disease in individuals.

Additionally, the heart disease prediction system can be used to raise awareness about the importance of maintaining a healthy lifestyle, including regular exercise, healthy eating, and stress management. This can encourage individuals to take proactive measures to reduce their risk of heart disease and adopt a healthier lifestyle.

Overall, the heart disease prediction system has the potential to improve the overall health outcomes of individuals and reduce the burden of heart disease on healthcare systems. However, it is important to ensure that the system is based on robust medical data and is used in conjunction with professional medical advice and care.

The prediction system was prepared using a Machine Learning model which was trained using Classification algorithms namely Support Vector Machine and the Naïve Bayes Classifiers.

The accuracy after training the model with Support Vector Machine was 91.22.

The accuracy after training the model with the Gaussian Naïve Bayes Classifier was 87.72 and the accuracy through Bernoulli Naïve Bayes Classifier was 85.96.

Therefore, the algorithm with the highest accuracy, i.e., SVM will be taken into application.

The model is taken into use by the user interface after connectivity with Flask for the prediction of the disease.

11.APPENDIX

Python

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant White space. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. Numpy NumPy is a library for the python programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim with contributions from several other developers. In 2005, Travis created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open source software and has many contributors.

Seaborn

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Flask

Flask is a Python web framework that is used to build web applications. It is a lightweight framework that is easy to use and allows developers to quickly build and deploy web applications. Flask is based on the WSGI (Web Server Gateway Interface) toolkit and is designed to be flexible and modular, allowing developers to use only the components they need for their specific application.

Flask provides a number of features that make it easy to develop web applications, such as routing, templating, and support for handling HTTP requests and responses. It also supports extensions that can be used to add additional functionality to the framework, such as support for databases, authentication, and user management.

Pickel

Pickle is a Python module that allows objects to be serialized and deserialized, which means they can be converted into a byte stream and stored in a file or sent over a network. This is useful when you need to save the state of an object so that it can be loaded later, or when you need to transfer data between different systems.

12. REFERENCES

1. https://ieeexplore.ieee.org/document/9734880
2. https://iopscience.iop.org/i
3. https://medium.com/analytics-vidhya/feature-selection-for-dimensionality-reduction-embedded-method-e05c74014aa
4. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd
5. https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction?resource=download
6. https://towardsdatascience.com/predicting-presence-of-heart-diseases-using-machine-learning-36f00f3edb2c
7. https://www.sciencedirect.com/science/article/pii/S2772442522000016
8. https://towardsdatascience.com/predicting-presence-of-heart-diseases-using-machine-learning-36f00f3edb2c
9. https://medium.com/analytics-vidhya/feature-selection-for-dimensionality-reduction-embedded-method-e05c74014aa
10. https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012046
11. Bashir S, Qamar U & Javed M Y (2014, November). A decision support framework for intelligent heart disease diagnosis. In International Conference on Information Society (i-Society 2014) (pp. 259-64). IEEE.